Barehands:
Implement-Free Interaction with a Wall-Mounted Display

Meredith Ringel
Computer Science Department
Brown University
Providence, RI 02912
mringel@cs.brown.edu

Henry Berg, Yuhui Jin, Terry Winograd
Computer Science Department
Stanford University
Stanford, CA 94305-9035
{hgberg, yhjin, winograd}@cs.stanford.edu

ABSTRACT
We describe Barehands, a free-handed interaction technique, in which the user can control the invocation of system commands and tools on a touch screen by touching it with distinct hand postures. Using behind-screen infrared (IR) illumination and a video camera with an IR filter, we enable a back-projected SMARTBoard (a commercially available, 61 '' x 47'' touch-sensing display) to identify and respond to several distinct hand postures. Barehands provides a natural, quick, implement-free method of interacting with large, wall-mounted interactive surfaces.

Keywords
Interaction technique, user interface, hand posture, infrared, image processing, region growing, SMARTBoard, Interactive Workspaces, touch interaction, interaction tool.

INTRODUCTION
As part of our project to develop a pervasive computing environment [6], we have created an *interactive workspace* which integrates a variety of devices, including laptops, PDAs, and large displays, both vertical (wall-mounted) and horizontal (tabletop). Our research focus is on providing integration at both the system and user-interaction levels, so that information and interfaces can be associated with a user and task rather than with a particular device or surface.

Barehands addresses the issue of effective interaction with large touch-sensitive surfaces by employing hand-posture recognition techniques.

The Overface
A key design criterion for our environment is to provide support on a variety of devices for existing modes of interaction with applications and standard GUI interfaces (e.g., Windows, PalmOS). We cannot expect real applications to be developed if they require special re-coding for use in our environment. At the same time, we want to support additional interactions that are not in current systems. These include:

- device augmentation (such as providing the equivalent of keyboard shortcuts for a non-keyboard touch screen)
- multi-device actions (such as bringing up a web page or application on a screen other than the one on which the interaction occurs, or using a pointing device on a laptop to control the cursor on a wall-screen)
- meta-screen actions (such as marking up the desktop display)
- space-global interactions (such as a global copy and paste, which allows objects to be moved from any device to any other).
- environment-control actions (such as turning projectors on and off and re-mapping their input sources),

The challenge is to add these in a consistent and uniform way that does not create confusion. Our approach is to create an *overface*, which provides affordances for these actions while also allowing the user to interact normally with the underlying interfaces.

In order to intermix overface and underlying interface actions, we need to distinguish the kind of action the user intends. This can be done with temporal modes (the user switches back and forth from overface to interface mode), with spatial modes (some areas of the display support overface actions while others support interface actions), or with physical modes (e.g., holding down a button, or using a mode-specific physical tool).

The problems created by temporal modes have been widely discussed in the HCI literature and are exacerbated in our multi-device multi-user setting. Keeping track of which surface is in which mode is a source of confusion and

error. Spatial modes work well when the activity can be separated (for example our controller for the lights and projectors, appears as a separate area on the screen), but do not work for actions that require being located on an application area (e.g., freehand markup or global copy and paste). Physical modes are used extensively in standard GUI environments, using devices such as the mouse, with its multiple buttons and keyboard modifiers (SHIFT, CTRL, etc.). Physical modes avoid the confusion of temporal modes because the user is directly aware of which mode is in effect, based on what is being pressed or held.

Bare hand touch screen interaction

A key component of our space is a wall with three adjacent back-projected SMARTBoards [8]: commercially available, wall-mounted, touch-sensitive displays. Users integrate work on individual devices (PDAs and laptops connected by a wireless LAN) with work by multiple people standing at the SMARTBoards. We normally display a standard Windows 2000 desktop on each of these boards or one combined desktop covering all three. Users are able to run any Windows applications that are desirable for their tasks.

In our experience over a year of using the boards, we recognized the strong appeal of direct hands-on manipulation without implements. Although the SMARTBoard provides a set of whiteboard tools (see Discussion below), users gravitated toward performing simple interactions by touching a finger to the board. In fact, they often tried fruitlessly to do the same on the adjacent front-projected wall, which had no touch sensors. We therefore decided to explore the possibilities for bare-hands mechanisms that could provide appropriate interaction for both interface and overface, and that did not require the user to hold separate implements.

Any touch on the SMARTBoard touch screen is interpreted by the standard software as a left mouse click. Simultaneous touch at multiple points is interpreted as a single mouse click with coordinates corresponding roughly to the center of the contact points. To achieve the affordances of ordinary workstations, and to add a limited form of overface, the SMARTBoard designers added both temporal and physical modes. Right-button click is implemented with a temporal mode that requires the user to press a physical button on a tray at the bottom, right-hand edge of the SMARTBoard. The next touch is then interpreted as a right mouse click. There is no way to indicate a position without a button click (as is used for affordances such as rollover tool-tips).

In addition to standard GUI functionality, the SMARTBoard and its accompanying software provide electronic whiteboard capabilities (freehand and geometrical figure markup on the display surface) using separate physical devices modeled after conventional whiteboard marker pens and erasers. There are a number of problems with these tools, and we wanted to see what could be done using a tool-free hands-only mechanism.

Although the touch screen hardware only identifies a single center-of-gravity point of contact, users can easily produce different hand postures, such as touching with a single finger, with multiple fingers, or with the side or palm of the hand. Barehands uses these postures as physical modes, allowing the touch to trigger actions depending on posture. A dynamic mapping allows any posture to be mapped to any of the available actions, including a basic OS interaction (left-button, right-button, etc.) or an overface interaction (markup, global actions, etc).

In our current stage of experimentation, we have not yet established the optimal mappings. To facilitate experimentation, we have developed a mapping interface that allows any posture to be mapped to any of the available actions, including whiteboard tools, Windows events (e.g. left or right mouse-click), and Windows commands (e.g. cut, copy, or paste). We can also extend the set of distinguishable actions further by mapping a posture to invoke Sensiva [7], a commercial gesture-recognition system, which allows an arbitrary number of different on-screen (2-dimensional) gestures to be mapped to desired actions.

Our research goal is to explore the space of hand postures and gestures to identify which ones are best suited to the different aspects of overface and interface.

PREVIOUS WORK

HoloWall [2], developed at the Sony Computer Science Laboratory in Tokyo, allows for object recognition from behind a translucent wall surface. IR illumination and a camera behind the wall are used to detect hands or inanimate objects touching the surface. Holding an object against the wall may trigger an associated system response (e.g. projecting a video onto the wall in response to touching it with a videocassette box). The HoloWall was a special-purpose device, not integrated with standard displays or interaction modes.

We have extended the visual object-tracking technique of the HoloWall not only to detect when and where a hand has touched the display, but to classify that touch into one of several categories. Currently, our system accurately identifies five distinct postures: one finger, two fingers, a vertical edge, a horizontal edge, and an entire palm. Users may map those five hand configurations to Windows commands (e.g. left mouse button, right mouse button, cut, copy, paste, etc.) or to the SMARTBoard's "whiteboard"

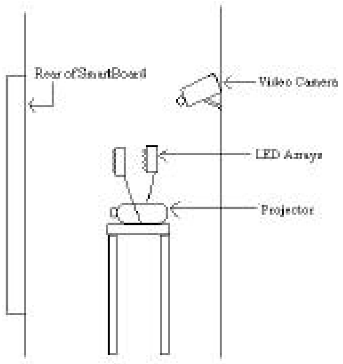

FIGURE 1: Projection, camera, and lighting setup, side view. The Infrared LED arrays are pulsed in coordination with the camera shutter to illuminate the rear of the board, including objects that reflect light by being near to its front side. The camera records the image for analysis.

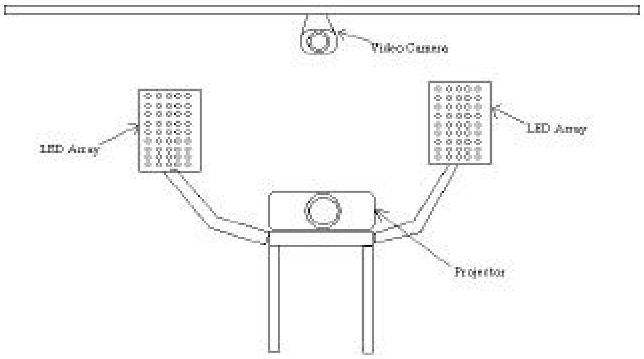

FIGURE 2: Front view of the setup as seen from rear of projection screen. Two side-displaced arrays are used to spread the lighting and to avoid a direct reflection off of the screen into the camera. Projector back-reflection is minimal since it is vertically displaced and the projector has an IR cutoff filter.

tools (e.g. pen, eraser, highlighter, rectangle tool, ellipse tool, line tool.) by invoking a selection tool, which is available in the Windows taskbar, at any time.

A two-fingered touch, for example, can be mapped to send a right mouse click This right-click message is then handled by the application within whose window the touch occurred. Similarly, by mapping Ctrl + C to one of the hand postures and Ctrl + V to another, a user can copy and paste an item simply by touching the screen twice (first with the copy posture, then with the paste posture).

Barehands allows us to easily integrate Sensiva [7], a freeware gesture-recognition tool, into our system. Sensiva provides functionality for recognizing and responding to different shapes drawn when the right mouse button is held down. Using Barehands to map a posture (two fingers, for example) to mean "right mouse button" allows us to quickly and intuitively augment the interaction space.

An alternative method for sensing hands on a display is the use of laser rangefinders. The recognition system produced by Stricken and Paradiso [5] was able to use hand position as a way of controlling audio output, but there was no attempt to use hand postures to distinguish actions, and the sensing method makes it difficult to distinguish postures.

Rehg and Kanade[3] used a camera in front of the display surface to analyze the 3-dimensional configuration of a hand. They used kinematic modeling techniques to model multiple degrees of freedom in hand configuration, which they used to control a 3-dimensional 6DOF mouse. Although the technique is much more general, its efficiency and accuracy are not in the range that would make it practical for our application.

Some issues related to working with large display surfaces have been addressed by the German National Research Center for Information Technology's (GMD) DynaWall [4] project, a part of their i-LAND endeavor. Their system takes advantage of the touch-sensitive nature of the SMARTBoard by translating certain pen motions into "shuffle" or "throw" messages, which move windows to a remote area of the display. Adopting a spatial mode mechanism, these motions must be made in a special "handle" portion of the icons displayed by the BEACH software framework [4], developed specifically for the i-LAND project. Touching a handle on the BEACH system's icons can also trigger a "take it" action, a hands-only version of a traditional cut and paste. Unlike DynaWall, whose underlying infrastructure is the specially tailored BEACH system, Barehands is built on top of a standard Windows 2000 operating platform.

BAREHANDS IMPLEMENTATION

In order to obtain images of a user's hand as it touches the display, we illuminate the rear of the screen with a pair of IR LED arrays, approximately 1.5m behind the board, each

FIGURE 3: A user touches the SMARTBoard with a "vertical edge" posture. The current recognized postures are one finger, two fingers, a vertical edge, a horizontal edge, and a palm.

FIGURE 4: Camera image of the user's hand on the rear of the board. After threshholding, the image is processed to produce characteristics of the area, such as vertical-to-horizontal ratio and perimeter-to-area ratio, which are used in posture recognition.

containing twelve rows of eight 16mw/str SLI-0308CP LEDs, pulsed at 200mA. Centered between the light sources approx 3m from the board, is a Marshall V-1070 video camera (resolution 811 x 510 pixels) with an IR filter, and an 8-80 mm motorized lens, specially adjusted to focus with IR light (Figures 1 and 2). A standard video digitizing card is used to capture the image from the camera.

IR light from the LED arrays is reflected off the rear of the screen and picked up by the camera. When a hand touches the front of the display, it reflects additional IR light, and is perceived by the camera as a region of increased intensity, as illustrated in Figures 3 and 4.

We initially constructed the LED arrays to provide continuous illumination and disabled the camera's shutter. This meant that the maximum continuous current allowed through the LEDs limited the amount of light. There was also no protection from ambient IR light. To solve these problems we modified the arrays to use an adjustable high-current pulse. We built a lighting control box that analyzes the video signal from the camera and extracts the shutter timing information necessary to synchronize the flash of light with the camera's electronic shutter. While we presently flash both arrays simultaneously, we designed the control box to allow independent field-by-field control of each array to support future work.

Image Processing

When the user touches the display, we first identify which pixels in the video image correspond to his or her hand by examining the pixels' grayscale intensity, and filtering out random noise (Figure 4).

We take advantage of the touch sensing of the SMARTBoard to reduce image-processing overhead. Although we receive a continual input from the video camera, we only analyze the frames of the video that occur immediately after we receive a "contact down" event from the SMARTBoard driver. This contact event is intercepted before it can be sent to Windows as a left mouse click event (the SMARTBoard software's default response to touch).

We are able to avoid processing the entire image by using the x- and y-coordinates associated with the contact-down event as a seed for region growing. Pixels within a fixed radius of the seed are examined and those with a grayscale intensity similar to that of the seed are included in the region. This process is repeated by exploring a fixed radius around each new pixel found to be in or near the region of contact. This allows for the detection of contact regions with or without discontinuities (a two-fingered contact, with the fingers spaced slightly apart, would result in a discontinuous region). Isolated pixels are eliminated, even if their intensity is above threshold.

Region characteristics such as height-to-width ratio, ratio of a region's area to the area of its bounding box, ratio of perimeter to area, and the presence or absence of gaps in the region are used to classify the image as either one finger, two fingers, a vertical edge, a horizontal edge, a palm, or unknown. Future research will investigate other

possible postures and recognition features, and we have experimented with using machine-learning systems for recognition, as well.

Action Mappings

Once the activated region has been classified as a particular hand posture, an appropriate action is produced, based on the mapping currently associated with that posture. If the posture is mapped to a whiteboard tool, we send messages to the SMARTBoard driver to change the current tool. If the posture is mapped to a Windows command, we send an appropriate combination of keyboard and/or mouse events.

An icon displayed in the Windows taskbar can be selected at any time to bring up a dialog box that allows the user to change any or all of the posture-to-action mappings. This dialog also allows the user to select whether a particular mapping is activated immediately after posture detection, or on the following touch.

In our initial experiments, we began with a purely physical mode, in which the hand posture at the beginning of a touch determined the interpretation of the motion throughout that touch. For example, if a two-finger touch mapped to the gesture recognizer, the user would touch down with two fingers and continue by drawing the gesture with them. We observed that for some mappings, this was awkward. For example, if a vertical-side-of-hand touch activates a drawing tool, such as a pen or highlighter, it is not convenient to then do the drawing with the hand held in that position. We have added a temporal mode, in which the posture of the initial touch determines the mapping that will be applied to the immediately following touch. So, for example, the user might touch with the side of the hand to indicate a drawing tool, and then use a single finger on the next touch to do the drawing. Because the two touches are immediately correlated and a change of cursor indicates the action that will be taken by the next touch, we have not found that this creates mode confusion.

EVALUATION

Informal initial observations of Barehands use affirm that our system provides a means of interacting with a large display that avoids the inconvenience of walking to a different area of the device to push a button or of grasping a physical interaction tool. By mapping gestures to commands such as copy and paste the user saves time and motions traditionally required for selecting those commands from menus.

System response time is conducive to real-time interaction: image-processing time averaged 13.37 ms for a mixed-posture set of 120 touches. Average posture-classification accuracy rates are above 90%; we expect that more uniform IR illumination in our remodeled configuration will allow for improved accuracy. Since there is immediate visual feedback on the posture that was recognized and opportunity for correction, the system should be useable with well below 100% accuracy, but it will take further experimentation to determine the best tradeoff between accuracy and the number of different postures recognized.

Systematic user studies are planned for the future, but could not be performed by the time of submission due to a remodeling of the Interactive Room, which required dismantling our lighting and camera setup soon after the initial experiments. Further evaluation of the effectiveness of the Barehands interaction technique will be carried out after remodeling is completed later this autumn.

DISCUSSION

The key assumption underlying Barehands is that for a significant class of users and interactions (in particular, the overface interactions), bare-handed interaction is better than using physical tools to determine the mapping from touch to action. An alternative approach is exemplified by the tools that come with the SMARTBoard, based on traditional colored marker pens and erasers. In order to annotate the screen with a colored mark, the user picks up the corresponding colored marker and draws with its tip (which is just a plastic cone). To erase, she picks up the eraser and uses it in the conventional way.

There are advantages to a tool-based and a tool-free approach:

- When there are obvious tools for the interactions, the mapping is natural. Picking up a red pen is an obvious way to draw a red mark.
- Tools can be shaped to be better suited to the task. Drawing with a pointed pen-tip is more accurate than drawing with a fingertip.
- Appropriate use of tools can avoid the need to augment the physical mode with temporal ones (our two-touch mode).

And there are complementary advantages for a tool-free approach

- Bare hands (unlike tools) can never be misplaced, are always available, and do not need to be handed off from user to user in a multi-user environment
- It is not necessary to invent a tool for actions that do not naturally suggest one (e.g., making an on-screen gesture).
- Specialized hardware is required for tool recognition and activation (of course there is also hardware for bare hands visual recognition).

In regard to the last point, the actual SMARTBoard tools are highly unsatisfactory for the kind of use we envision. They are housed in a tray equipped with detectors to sense the presence or absence of each tool. When a tool is

removed from the board's tray, all subsequent touches to the board (by any device, including bare fingers) are interpreted in a mode associated with the tool that was last picked up, until the tool is replaced in its tray or another one is picked up.

This pseudo-physical mode (it has the intended physical mapping when the user picks up only one tool at a time and then replaces it) works relatively well for the single-whiteboard-replacement applications that are the primary target for the device. However, confusion often results, for example due to picking up both a pen and the eraser (after which touches by either are interpreted according to whichever was the last picked up). In our environment, further confusion results both from the multiple-device context (each of the three boards has its own tray of implements and associated sensors but people pick up tool on one board and then use it on another), and the mixing of whiteboard actions (such as sketching) with application interaction modes that are not part of simple pen-on-whiteboard activity.

These flaws could be corrected by using some different mechanism for tool sensing, and one of our long-term research questions is understanding the criteria that make tool-based and bare-hand interaction appropriate for different kinds of situations and actions.

Future Work

Lighting arrangements have been the greatest technical challenge in the implementation of Barehands. The setup shown in Figures 1 and 2, employing two arrays of IR LEDs, provides non-uniform lighting, as the light from each LED spreads out over a 30 degree arc. The central regions of the board are well lit, but the outer edges remain dark. We have tried unsuccessfully to solve this problem with various conventional lenses and filters. The uneven light results in lower recognition accuracy for hand postures outside of the display's central area.

When we set up our lighting again, after the Interactive Room remodeling is complete, we plan to experiment with new arrangements of IR light sources to achieve more uniform illumination. Since each source is composed of many LEDs, they can be arranged into other geometries, such as linear strings rather than concentrated arrays. We will also experiment with using alternating lighting directions to improve recognition. We are planning to experiment with lighting schemes where successive frames of video are lit in different ways and from different directions, enabling more sophisticated image analysis.

More uniform lighting will facilitate more accurate image processing, which will enable us to expand our current set of five recognized postures. Although the space of recognizable postures is large, many of the theoretically possible ones will be awkward or uncomfortable. In our user experiments we will look for postures that are ergonomically appropriate and that have natural mappings to actions (e..g., using the side of the hand as an eraser).

In addition to recognizing static hand postures, we plan to extend the system to recognize postural motions (such as "plucking" an item from the board), which will extend the space of potentially meaningful actions. We also can easily extend the visual recognition algorithms to identify objects other than hands. A physical object of a known shape can be touched to the board and manipulated to achieve a desired tool-specific interaction. Although our general approach has emphasized implement-free interaction, there may be places where specialized tools are appropriate in combination with the Barehands interaction.

ACKNOWLEDGEMENTS

We are indebted to all of the people in our research group who have worked to create the interactive workspace environment. Winfield Hill at the Rowland Institute for Science provided invaluable assistance in the design of the pulsed lighting system. Doug Hill at SMART Technologies provided valuable information and assistance. Special thanks to Dan Morris and François Guimbretière for improvements to this paper.

REFERENCES

1. Geißler, Jörg. Shuffle, throw or take it! Working Efficiently with an Interactive Wall. CHI '98 Summary, ACM Press, 265-266.

2. Matsushita, Nobuyuki, and Jun Rekimoto. HoloWall: Designing a Finger, Hand, Body, and Object Sensitive Wall. CHI '97 (Banff, Canada, October 1997), ACM Press, 209-210.

3. Rehg, J. and T. Kanade, Digiteyes: Viesion-Based Hand Tracking for Human-Computer Interaction. Proc. Of the Workshop on Motion of Non-rigid and Articulated Objects, Austin Texas, November 1994, IEEE Computer Society Press, pages 16-22.

4. Streitz, Norbert A., et. al. i-LAND: An Interactive Landscape for Creativity and Innovation. CHI '99 (Pittsburgh, PA, May 1999), ACM Press, 120-127.

5. Stricken, Joshua, and Joseph Paradiso, Tracking Hands Above Large Interactive Aurfaces with a Low-Cost Scanning Laser Rangefinder, CHI98 Summary, 231-232.

6. See http://graphics.stanford.edu/projects/iwork

7. See http://www.sensiva.com

8. See http://www.smarttech.com